

1 What is claimed is:

2

3 1. A method of operating a network file server for providing clients with concurrent

4 write access to a file, the method comprising the network file server responding to a

5 concurrent write request from a client by:

6 (a) obtaining a lock for the file; and then

7 (b) preallocating a metadata block for the file; and then

8 (c) releasing the lock for the file; and then

9 (d) asynchronously writing to the file; and then

10 (e) obtaining the lock for the file; and then

11 (f) committing the metadata block to the file; and then

12 (g) releasing the lock for the file.

13

14 2. The method as claimed in claim 1, wherein the file includes a hierarchy of blocks

15 including an inode block of metadata, data blocks of file data, and indirect blocks of

16 metadata, and wherein the metadata block for the file is an indirect block of metadata.

17

18 3. The method as claimed in claim 2, which includes copying data from an original

19 indirect block of the file to the metadata block for the file, the original indirect block of

20 the file having been shared between the file and a read-only version of the file.

21

22 4. The method as claimed in claim 1, which includes concurrent writing for more

23 than one client to the metadata block for the file.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

5. The method as claimed in claim 1, wherein the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon failing to find an indication of a conflict with a concurrent write to the new block, preallocating the new block, copying at least a portion of the original block of the file to the new block, and performing the partial write to the new block.

6. The method as claimed in claim 1, wherein the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block.

7. The method as claimed in claim 6, which includes placing a request for the partial write in a partial write wait queue upon finding an indication of a conflict with a concurrent write to the new block, and performing the partial write upon servicing the partial write wait queue.

1 8. The method as claimed in claim 1, wherein the asynchronously writing to the file
2 includes checking an input-output list for a conflicting prior concurrent access to the file,
3 and upon finding a conflicting prior concurrent access to the file, suspending the
4 asynchronous writing to the file until the conflicting prior concurrent access to the file is
5 no longer conflicting.

6

7 9. The method as claimed in claim 8, wherein the suspending of the asynchronous
8 writing to the file until the conflicting prior concurrent access is no longer conflicting
9 provides a sector-level granularity of byte range locking for concurrent write access to
10 the file.

11

12 10. The method as claimed in claim 1, wherein the metadata block for the file is
13 committed by writing the metadata block to a log in storage of the network file server.

14

15 11. The method as claimed in claim 1, which includes gathering together preallocated
16 metadata blocks for a plurality of client write requests to the file, and committing
17 together the preallocated metadata blocks for the plurality of client write requests to the
18 file by obtaining the lock for the file, committing the gathered preallocated metadata
19 blocks for the plurality of client write requests to the file, and then releasing the lock for
20 the file.

21

22 12. The method as claimed in claim 1, which includes checking whether a previous
23 commit is in progress after asynchronously writing to the file and before obtaining the

1 lock for the file for committing the metadata block to the file, and upon finding that a
2 previous commit is in progress, placing a request for committing the metadata block to
3 the file on a staging queue for the file.
4

5 13. A method of operating a network file server for providing clients with concurrent
6 write access to a file, the method comprising the network file server responding to a
7 concurrent write request from a client by:

8 (a) preallocating a block for the file; and then

9 (b) asynchronously writing to the file; and then

10 (c) committing the block to the file;

11 wherein the asynchronous writing to the file includes a partial write to a new
12 block that has been copied at least in part from an original block of the file, and wherein
13 the method includes checking a partial block conflict queue for a conflict with a
14 concurrent write to the new block, and upon finding an indication of a conflict with a
15 concurrent write to the new block, waiting until resolution of the conflict with the
16 concurrent write to the new block, and then performing the partial write to the new block.
17

18 14. The method as claimed in claim 13, wherein the method includes placing a
19 request for the partial write in a partial write wait queue upon finding an indication of a
20 conflict with a concurrent write to the new block, and performing the partial write upon
21 servicing the partial write wait queue.
22

1 15. A method of operating a network file server for providing clients with concurrent
2 write access to a file, the method comprising the network file server responding to a
3 concurrent write request from a client by:

4 (a) preallocating a metadata block for the file; and then

5 (b) asynchronously writing to the file; and then

6 (c) committing the metadata block to the file;

7 wherein the method includes gathering together preallocated metadata blocks for
8 a plurality of client write requests to the file, and committing together the preallocated
9 metadata blocks for the plurality of client write requests to the file by obtaining a lock for
10 the file, committing the gathered preallocated metadata blocks for the plurality of client
11 write requests to the file, and then releasing the lock for the file.

12
13 16. The method as claimed in claim 15, which includes checking whether a previous
14 commit is in progress after asynchronously writing to the file and before obtaining the
15 lock for the file for committing the block to the file, and upon finding that a previous
16 commit is in progress, placing a request for committing the metadata block to the file on
17 a staging queue for the file.

18
19 17. The method as claimed in claim 15, wherein the network file server includes disk
20 storage containing a file system, and a file system cache storing data of blocks of the file,
21 and the method includes the network file server responding to concurrent write requests
22 by writing new data for specified blocks of the file to the disk storage without writing the

1 new data for the specified blocks of the file to the file system cache, and invalidating the
2 specified blocks of the file in the file system cache.

3

4 18. The method as claimed in claim 17, which includes the network file server
5 responding to read requests for file blocks not found in the file system cache by reading
6 the file blocks from the file system in disk storage and then checking whether the file
7 blocks have become stale due to concurrent writes to the file blocks, and writing to the
8 file system cache a file block that has not become stale, and not writing to the file system
9 cache a file block that has become stale.

10

11 19. The method as claimed in claim 18, which includes the network file server
12 checking a read-in-progress flag for a file block upon finding that the file block is not in
13 the file system cache, and upon finding that the read-in-progress flag indicates that a prior
14 read of the file block is in progress from the file system in the disk storage, waiting for
15 completion of the prior read of the file block from the file system in the disk storage, and
16 then again checking whether the file block is in the file system cache.

17

18 20. The method as claimed in claim 18, which includes the network file server setting
19 a read-in-progress flag for a file block upon finding that the file block is not in the file
20 system cache and then beginning to read the file block from the file system in disk
21 storage, clearing the read-in-progress flag upon writing to the file block on disk, and
22 inspecting the read-in-progress flag to determine whether the file block has become stale
23 due a concurrent write to the file block.

1

2 21. The method as claimed in claim 18, which includes the network file server
3 maintaining a generation count for each read of a file block from the file system in the
4 disk storage in response to a read request for a file block that is not in the file system
5 cache, and checking whether a file block having been read from the file system in the
6 disk storage has become stale by checking whether the generation count for the file block
7 having been read from the file system is the same as the generation count for the last read
8 request for the same file block.

9

10 22. The method as claimed in claim 15, which includes processing multiple
11 concurrent read and write requests by pipelining the requests through a first processor
12 and a second processor, the first processor performing metadata management for the
13 multiple concurrent read and write requests, and the second processor performing
14 asynchronous reads and writes for the multiple concurrent read and write requests.

15

16 23. The method as claimed in claim 15, which includes serializing the reads by
17 delaying access for each read to a block that is being written to by a prior, in-progress
18 write until completion of the write to the block that is being written to by the prior, in-
19 progress write.

20

21 24. The method as claimed in claim 15, which includes serializing the writes by
22 delaying access for each write to a block that is being accessed by a prior, in-progress

1 read or write until completion of the read or write to the block that is being accessed by
2 the prior, in-progress read or write.

3

4 25. A method of operating a network file server for providing clients with concurrent
5 read and write access to a file, the method comprising the network file server responding
6 to a concurrent write request from a client by:

7 (a) preallocating a metadata block for the file; and then

8 (b) asynchronously writing to the file; and then

9 (c) committing the metadata block to the file;

10 wherein the network file server includes disk storage containing a file system, and
11 a file system cache storing data of blocks of the file, and the method includes the network
12 file server responding to concurrent write requests by writing new data for specified
13 blocks of the file to the disk storage without writing the new data for the specified blocks
14 of the file to the file system cache, and invalidating the specified blocks of the file in the
15 file system cache, and

16 which includes the network file server responding to read requests for file blocks
17 not found in the file system cache by reading the file blocks from the file system in disk
18 storage and then checking whether the file blocks have become stale due to concurrent
19 writes to the file blocks, and writing to the file system cache a file block that has not
20 become stale, and not writing to the file system cache a file block that has become stale.

21

22 26. The method as claimed in claim 25, which includes the network file server
23 checking a read-in-progress flag for a file block upon finding that the file block is not in

1 the file system cache, and upon finding that the read-in-progress flag indicates that a prior
2 read of the file block is in progress from the file system in the disk storage, waiting for
3 completion of the prior read of the file block, and then again checking whether the file
4 block is in the file system cache.

5
6 27. The method as claimed in claim 25, which includes the network file server setting
7 a read-in-progress flag for a file block upon finding that the file block is not in the file
8 system cache and then beginning to read the file block from the file system in disk
9 storage, clearing the read-in-progress flag upon writing to the file block on disk, and
10 inspecting the read-in-progress flag to determine whether the file block has become stale
11 due a concurrent write to the file block.

12
13 28. The method as claimed in claim 25, which includes the network file server
14 maintaining a generation count for each read of a file block from the file system in the
15 disk storage in response to a read request for a file block that is not in the file system
16 cache, and checking whether a file block having been read from the file system in the
17 disk storage has become stale by checking whether the generation count for the file block
18 having been read from the file system is the same as the generation count for the last read
19 request for the same file block.

20
21 29. A method of operating a network file server for providing clients with concurrent
22 read and write access to a file, the method comprising the network file server responding
23 to a concurrent write request from a client by:

1 (a) preallocating a metadata block for the file; and then
2 (b) asynchronously writing to the file; and then
3 (c) committing the metadata block to the file;
4 wherein the method includes processing multiple concurrent read and write
5 requests by pipelining the requests through a first processor and a second processor, the
6 first processor performing metadata management for the multiple concurrent read and
7 write requests, and the second processor performing asynchronous reads and writes for
8 the multiple concurrent read and write requests.

9
10 30. The method as claimed in claim 29, which includes serializing the reads by
11 delaying access for each read to a block that is being written to by a prior, in-progress
12 write until completion of the write to the block that is being written to by the prior, in-
13 progress write.

14
15 31. The method as claimed in claim 29, which includes serializing the writes by
16 delaying access for each write to a block that is being accessed by a prior, in-progress
17 read or write until completion of the read or write to the block that is being accessed by
18 the prior, in-progress read or write.

19
20 32. A method of operating a network file server for providing clients with concurrent
21 write access to a file, the method comprising the network file server responding to a
22 concurrent write request from a client by executing a write thread, execution of the write
23 thread including:

- 1 (a) obtaining an allocation mutex for the file; and then
- 2 (b) preallocating new metadata blocks that need to be allocated for writing to the
- 3 file; and then
- 4 (c) releasing the allocation mutex for the file; and then
- 5 (d) issuing asynchronous write requests for writing to the file;
- 6 (e) waiting for callbacks indicating completion of the asynchronous write
- 7 requests; and then
- 8 (f) obtaining the allocation mutex for the file; and then
- 9 (g) committing the preallocated metadata blocks; and then
- 10 (h) releasing the allocation mutex for the file.
- 11
- 12 33. A network file server comprising storage for storing a file, and at least one
- 13 processor coupled to the storage for providing clients with concurrent write access to the
- 14 file, wherein the network file server is programmed for responding to a concurrent write
- 15 request from a client by:
- 16 (a) obtaining a lock for the file; and then
- 17 (b) preallocating a metadata block for the file; and then
- 18 (c) releasing the lock for the file; and then
- 19 (d) asynchronously writing to the file; and then
- 20 (e) obtaining the lock for the file; and then
- 21 (f) committing the metadata block to the file; and then
- 22 (g) releasing the lock for the file.
- 23

1 34. The network file server as claimed in claim 33, wherein the file includes a
2 hierarchy of blocks including an inode block of metadata, data blocks of file data, and
3 indirect blocks of metadata, and wherein the metadata block for the file is an indirect
4 block of metadata.

5
6 35. The network file server as claimed in claim 34, which is programmed for copying
7 data from an original indirect block of the file to the metadata block for the file, the
8 original indirect block of the file having been shared between the file and a read-only
9 version of the file.

10
11 36. The network file server as claimed in claim 33, which is programmed for
12 concurrent writing for more than one client to the metadata block for the file.

13
14 37. The network file server as claimed in claim 33, which includes a partial block
15 conflict queue for indicating a concurrent write to a new block that is being copied at
16 least in part from an original block of the file, and wherein the network file server is
17 programmed to respond to a client request for a partial write to the new block by
18 checking the partial block conflict queue for a conflict, and upon failing to find an
19 indication of a conflict, preallocating the new block, copying at least a portion of the
20 original block of the file to the new block, and performing a partial write to the new
21 block.

22

1 38. The network file server as claimed in claim 33, which includes a partial block
2 conflict queue for indicating a concurrent write to a new block that is being copied at
3 least in part from an original block of the file, and wherein the network file server is
4 programmed to respond to a client request for a partial write to the new block by
5 checking the partial block conflict queue for a conflict, and upon finding an indication of
6 a conflict, waiting until resolution of the conflict with the concurrent write to the new
7 block, and then performing the partial write to the new block.

8
9 39. The network file server as claimed in claim 38, which includes a partial write wait
10 queue, and wherein the network file server is programmed for placing a request for the
11 partial write in the partial write wait queue upon finding an indication of a conflict, and
12 performing the partial write upon servicing the partial write wait queue.

13
14 40. The network file server as claimed in claim 33, which is programmed for
15 maintaining an input-output list of concurrent reads and writes to the file, and when
16 writing to the file, for checking the input-output list for a conflicting prior concurrent
17 read or write access to the file, and upon finding a conflicting prior concurrent read or
18 write access to the file, suspending the asynchronous writing to the file until the
19 conflicting prior concurrent read or write access to the file is no longer conflicting.

20
21 41. The network file server as claimed in claim 40, wherein the suspending of the
22 asynchronous writing to the file until the conflicting prior concurrent read or write access

1 to the file is no longer conflicting provides a sector-level granularity of byte range
2 locking for concurrent write access to the file.

3

4 42. The network file server as claimed in claim 33, which is programmed for
5 maintaining an input-output list of concurrent reads and writes to the file, and when
6 reading from the file, for checking the input-output list for a conflicting prior concurrent
7 write access to the file, and upon finding a conflicting prior concurrent write access to the
8 file, suspending the reading to the file until the conflicting prior concurrent write access
9 to the file is no longer conflicting.

10

11 43. The network file server as claimed in claim 42, wherein the suspending of the
12 reading to the file until the conflicting prior concurrent write access to the file is no
13 longer conflicting provides a sector-level granularity of byte range locking for concurrent
14 read access to the file.

15

16 44. The network file server as claimed in claim 33, which is programmed for
17 committing the metadata block for the file by writing the metadata block to a log in the
18 storage.

19

20 45. The network file server as claimed in claim 33, which is programmed for
21 gathering together preallocated metadata blocks for a plurality of client requests for write
22 access to the file, and committing together the preallocated metadata blocks for the
23 plurality of client requests for access to the file by obtaining the lock for the file,

1 committing the gathered preallocated metadata blocks for the plurality of client requests
2 for write access to the file, and then releasing the lock for the file.

3

4 46. The network file server as claimed in claim 33, which includes a staging queue
5 for the file, and which is programmed for checking whether a previous commit is in
6 progress after asynchronously writing to the file and before obtaining the lock for the file
7 for committing the metadata block to the file, and upon finding that a previous commit is
8 in progress, placing a request for committing the metadata block to the file on the staging
9 queue for the file.

10

11 47. A network file server comprising storage for storing a file, and at least one
12 processor coupled to the storage for providing clients with concurrent write access to the
13 file, wherein the network file server is programmed for responding to a concurrent write
14 request from a client by:

15 (a) preallocating a block for the file; and then

16 (b) asynchronously writing to the file; and then

17 (c) committing the block to the file;

18 wherein the network file server includes a partial block conflict queue for indicating a
19 concurrent write to a new block that is being copied at least in part from an original block
20 of the file, and wherein the network file server is programmed for responding to a client
21 request for a partial write to the new block by checking the partial block conflict queue
22 for a conflict, and upon finding an indication of a conflict, waiting until resolution of the

1 conflict with the concurrent write to the new block of the file, and then performing the
2 partial write to the new block of the file.

3

4 48. The network file server as claimed in claim 47, which includes a partial write wait
5 queue, and wherein the network file server is programmed for placing a request for the
6 partial write in the partial write wait queue upon finding an indication of a conflict, and
7 performing the partial write upon servicing the partial write wait queue.

8

9 49. A network file server comprising storage for storing a file, and at least one
10 processor coupled to the storage for providing clients with concurrent write access to the
11 file, wherein the network file server is programmed for responding to a concurrent write
12 request from a client by:

13 (a) preallocating a metadata block for the file; and then

14 (b) asynchronously writing to the file; and then

15 (c) committing the metadata block to the file;

16 wherein the network file server is programmed for gathering together preallocated
17 metadata blocks for a plurality of client write requests to the file, and committing
18 together the preallocated metadata blocks for the plurality of client write requests to the
19 file by obtaining a lock for the file, committing the gathered preallocated metadata blocks
20 for the plurality of client write requests to the file, and then releasing the lock for the file.

21

22 50. The network file server as claimed in claim 49, which is programmed for
23 checking whether a previous commit is in progress after asynchronously writing to the

1 file and before obtaining the lock for the file for committing the metadata block to the
2 file, and upon finding that a previous commit is in progress, placing a request for
3 committing the metadata block to the file on a staging queue for the file.
4

5 51. A network file server comprising disk storage containing a file system, and a file
6 system cache storing data of blocks of a file in the file system, wherein the network file
7 server is programmed for responding to a concurrent write request from a client by:

8 (a) preallocating a metadata block for the file; and then

9 (b) asynchronously writing to the file; and then

10 (c) committing the metadata block to the file;

11 wherein the network file server is further programmed for responding to
12 concurrent write requests by writing new data for specified blocks of the file to the disk
13 storage without writing the new data for the specified blocks of the file to the file system
14 cache, and invalidating the specified blocks of the file in the file system cache, and

15 wherein the network file server is programmed for responding to concurrent read
16 requests for file blocks not found in the file system cache by reading the file blocks from
17 the file system in disk storage and then checking whether the file blocks have become
18 stale due to concurrent writes to the file blocks, and writing to the file system cache a file
19 block that has not become stale, and not writing to the file system cache a file block that
20 has become stale.
21

22 52. The network file server as claimed in claim 51, which is programmed for
23 checking a read-in-progress flag for a file block upon finding that the file block is not in

1 the file system cache, and upon finding that the read-in-progress flag indicates that a prior
2 read of the file block is in progress from the file system in the disk storage, waiting for
3 completion of the prior read of the file block, and then again checking whether the file
4 block is in the file system cache.

5
6 53. The network file server as claimed in claim 51, which is programmed for setting a
7 read-in-progress flag for a file block upon finding that the file block is not in the file
8 system cache and then beginning to read the file block from the file system in disk
9 storage, clearing the read-in-progress flag upon writing to the file block on disk, and
10 inspecting the read-in-progress flag to determine whether the file block has become stale
11 due a concurrent write to the file block.

12
13 54. The network file server as claimed in claim 51, which is programmed for
14 maintaining a generation count for each read of a file block from the file system in the
15 disk storage in response to a read request for a file block that is not in the file system
16 cache, and checking whether a file block having been read from the file system in the
17 disk storage has become stale by checking whether the generation count for the file block
18 having been read from the file system is the same as the generation count for the last read
19 request for the same file block.

20

21

22

1 55. A network file server comprising storage for storing a file, a first processor
2 coupled to the storage for managing metadata of the file, and a second processor for
3 performing asynchronous reads and writes to the file, wherein the network file server is
4 programmed for responding to a concurrent write request from a client by:
5 (a) preallocating a metadata block for the file; and then
6 (b) asynchronously writing to the file; and then
7 (c) committing the metadata block to the file;
8 wherein the network file server is programmed for processing multiple concurrent
9 read and write requests by pipelining the multiple concurrent read and write requests
10 through the first and second processors.

11
12 56. The network file server as claimed in claim 55, which is programmed for
13 serializing the reads to the file by delaying access for each read to a block of the file that
14 is being written to by a prior, in-progress write until completion of the write to the block
15 of the file that is being written to by the prior, in-progress write.

16
17 57. The network file server as claimed in claim 55, which is programmed for
18 serializing the writes to blocks of the file by delaying access for each write to a block of
19 the file that is being accessed by a prior, in-progress read or write.

20
21 58. A network file server comprising storage for storing a file, and at least one
22 processor coupled to the storage for providing clients with concurrent write access to the

1 file, wherein the network file server is programmed with a write thread for responding to
2 a concurrent write request from a client by:

3 (a) obtaining an allocation mutex for the file; and then

4 (b) preallocating new metadata blocks that need to be allocated for writing to the
5 file; and then

6 (c) releasing the allocation mutex for the file; and then

7 (d) issuing asynchronous write requests for writing to the file;

8 (e) waiting for callbacks indicating completion of the asynchronous write
9 requests; and then

10 (f) obtaining the allocation mutex for the file; and then

11 (g) committing the preallocated metadata blocks; and then

12 (h) releasing the allocation mutex for the file.

13
14 59. The network file server as claimed in claim 58, which includes an uncached write
15 interface, a file system cache and a cached read-write interface, and wherein the
16 uncached write interface bypasses the file system cache for sector-aligned write
17 operations.

18
19 60. The network file server as claimed in claim 59, wherein the network file server is
20 programmed to invalidate cache blocks in the file system cache including sectors being
21 written to by the cached read-write interface.

22

1 61. A network file server comprising storage for storing a file, and at least one
2 processor coupled to the storage for providing clients with concurrent write access to the
3 file, wherein the network file server is programmed for responding to a concurrent write
4 request from a client by:

- 5 (a) preallocating a block for writing to the file;
- 6 (b) asynchronously writing to the file; and then
- 7 (c) committing the preallocated block;

8 wherein the network file server also includes an uncached write interface, a file system
9 cache, and a cached read-write interface, wherein the uncached write interface bypasses
10 the file system cache for sector-aligned write operations, and the network file server is
11 programmed to invalidate cache blocks in the file system cache including sectors being
12 written to by the cached read-write interface.

13

14